CSE440: Natural Language Processing 2

Project Report

Topic: Online Sexism Detection

Submitted by:

Section: 1

Munia Shaheen (20101050)

Akib Zabed Ifti (20101113)

Junaed Hossain (20101196)

Submitted to: Mr. Farig Yousuf Sadeque

Date of Submission: 11/5/2023

Introduction

Online sexism is becoming a bigger issue. Automated tools are increasingly frequently used to detect and evaluate sexist content on a large scale, however most of them just classify content into general, high-level categories without providing any further details. The capacity to identify sexist content and to explain why it is sexist enhances the interpretability, trust, and comprehension of the choices made by automated systems, giving users and moderators more control. In order to identify racism and sexism, we utilized a variety of models in this study, including Logistic Regression, K Neighbors Classifier, Bernoulli Naive Bayes, Multinomial Naive Bayes, SVM, Random Forest, and Deep Neural Network based classifiers. We will use the F1-score for evaluation and pick the best model for our task.

Dataset

We utilized a dataset of 14000 rows and 5 columns of text messages. Review id, text, label sexism, category, and vector were all contained in the columns. Since detecting sexism was a binary classification problem, we simply needed the review id and text fields.

Methodology

Data cleaning:

Replace contractions: A contraction in the English language is a word or phrase that has been abbreviated by omitting one or more letters, such as "I'm" in place of "I am". The contractions can be divided ("I'm" to "I "+"'m") or restored to their original form ("I'm" to "I am"). In my experience, the latter is more effective because sub-words like "'m" are more difficult to find a word embedding for.

Removing the punctuation: Without commas, brackets, or other punctuation, the sentences should be written. A list of punctuation characters is provided by the Python constant string.punctuation. This list will be used to remove all punctuation from our text.

Remove html tags: We must remove the html tags to make sure there is no invalid data in the dataset.

Single character removal: When we remove apostrophes from the word "Mark's", the apostrophe is replaced by an empty space. Hence, we are left with the single character "s" that we are removing here.

Remove multiple spaces: Next, we remove all the single characters and replace it by a space which creates multiple spaces in our text. Finally, we remove the multiple spaces from our text as well.

Dropping columns: Review id, label category, and vector columns were all dropped as de-

tecting sexism was a binary classification problem and these columns were unnecessary.

Tokenize and padding:

In the majority of NLP tasks, each word in the text must be represented by an integer value (index) before being fed to any model. The text will be changed into a series of integer values in this manner. We will use vectorizer to convert a collection of text documents to a matrix of token counts for the basic ML models.
The text can be tokenized using a Keras API for the DL models. Each different word's frequency is determined by the Keras tokenizer, which then arranges the words according to that frequency. By converting each text into either a series of integers (each integer representing the index of a token in a dictionary) or a vector with a binary coefficient for each token depending on word count and based on tf-idf, this enables the vectorization of a corpus of texts.

All your input sequences to the model need to have the same length. In order to achieve that, we can use a function that pads the short sequences with zeros (options are 'pre' or 'post' which pads either before or after each sequence). This function transforms a list (of length num_samples) of sequences (lists of integers) into a 2D Numpy array of shape (num_samples, num_timesteps). we are using the max-len parameter of the function. While using the max-len parameter, the pad_sequences function shows the array of the same length we have defined in the value of max-len arguments.

Word embedding:

Now that we have tokenized the text, we use GloVe pretrained vectors. Word embeddings is a way to represent words with similar meaning to have a similar representation. Using word embedding through GloVe, we can have a decent performance with models with even relatively small label training sets.

Models:

First, we splitted our data into train and test data and used Logistic Regression, K Neighbors Classifier, Bernoulli Naive Bayes, Multinomial Naive Bayes, SVM and Random Forest.
Then using the embedding matrix based on GloVe vectors, we can design a model and pre-train the embedding layer with it. The embedded layer is the top layer. The Keras embedding layer is a versatile layer that can be utilized with or without weights that have already been trained. In that instance, the Embedding layer will learn an embedding for each word in the training dataset after being initialized with random weights. In this experiment, the embedding layer's weights will be set to the embedding matrix derived from the GloVe pre-trained vectors.This learning is transferable. The "trainable" parameter in the Embedding layer can be set to True if we want to fine-tune the word embedding or False if we don't want the embedding weights to be updated. In this case, we set it to False. A

regularization-focused Dropout layer, an LSTM, GRU, and bidirectional LSTM are placed after the Embedding layer. Our validation split for each deep learning model was 0.2, batch size was 128 and epochs were 50. A dense layer with sigmoid activation transforms the output of the preceding layers into 0 or 1 (sexist or not sexist), and we employed the Adam optimizer in our models.

Result

To evaluate the effectiveness of our proposed sexism detection approach, we conducted experiments on a benchmark dataset consisting of 14,000 texts annotated with sexism labels by human experts. First, we evaluated the efficacy of various sexism detection machine learning algorithms, such as Logistic Regression, K Neighbors Classifier, Bernoulli Naive Bayes, Multinomial Naive Bayes, SVM, Random Forest. Our tests revealed that Random Forest outperformed the other models, with an F1 score of 0.89. We further compared the performance of the LSTM, GRU, Bi-LSTM model with several baseline methods, including logistic regression, decision tree, and support vector machine (SVM). Our experiments showed that the LSTM model outperformed all baseline methods in terms of F1 score of 0.88.
Overall, our experiments demonstrate the effectiveness of our proposed approach for detecting sexism in social media, and highlight the importance of using a combination of lexical, syntactic, semantic, and user features for achieving high performance.

Models	***F1-Score***
Logistic Regression	*0.8818*
KNN	*0.8557*
BernoulliNB	*0.8679*
MultinomialNB	*0.8771*
SVM	*0.8621*
Random Forest	*0.8935*
LSTM	*0.8852*
GRU	*0.8562*
Bi-LSTM	*0.8654*

E. Reference

1. *CodaLab - Competition*. (n.d.). https://codalab.lisn.upsaclay.fr/competitions/7124
2. Sepidafs. (2020). Text Classification with GloVe, LSTM/GRU(99% acc). *Kaggle*. https://www.kaggle.com/code/sepidafs/text-classification-with-glove-lstm-gru-99-acc/notebook
3. The AI & DS Channel. (2022, March 31). *Twitter Hate speech detection using machine learning | Machine Learning Project 9* [Video]. YouTube. https://www.youtube.com/watch?v=TLbb1UbU0aQ
4. Taha Junaid. (2020, September 22). *NLP-Hate Speech Detection using LSTM in Tensorflow* [Video]. YouTube. https://www.youtube.com/watch?v=2qnXmB65Ino
5. Rithesh Sreenivasan. (2020, September 11). *Text Classification using LSTM on Amazon Review Dataset with TensorFlow 2.1 #nlp #tutorial* [Video]. YouTube. https://www.youtube.com/watch?v=DUZn8hMLnbI
6. Nissa, N. K. (2022, August 23). Text Messages Classification using LSTM, Bi-LSTM, and GRU. *Medium*. https://medium.com/mlearning-ai/the-classification-of-text-messages-using-lstm-bi-lstm-and-gru-f79b207f90ad#: :t ext=A%20Gated%20Recurrent%20Unit%2C%20or,train%20than%20their%20LSTM%20counterparts.